Amendments to the Claims

Please replace the claims as follows:

Please cancel claims 6, 13, 21, and 28 without prejudice.

1. (currently amended) A computer-implemented method for verifying at runtime an invariant property of a data structure of a computer program, comprising:

automatically generating a first code segment that verifies a runtime value of the data structure is consistent with the invariant property in response to an annotation of the data structure that defines the invariant property of the data structure, wherein the invariant property is a range of addresses specified in the annotation in source code of the computer program;

comparing the runtime value of the data structure with the invariant property during execution of the program via execution of the first code segment, and verifying that the runtime value of the data structure is within the range of addresses; and

performing a programmed action if the runtime value is inconsistent with the invariant property.

2. (currently amended) The method of claim 1, wherein the range of addresses ~~invariant property~~ is a range of data addresses ~~and further comprising verifying that the runtime value of the data structure is within a range of data addresses specified in source code of the computer program~~.

3. (currently amended) The method of claim 1, wherein the range of addresses ~~invariant property~~ is a range of data addresses and further comprising:

automatically generating during compilation a valid data address range including an upper bound and a lower bound for the range of data addresses, wherein the source code of the computer program does not include a specification of the upper bound and lower bound; and

verifying that the runtime value of the data structure is within the valid data address range.

4. (currently amended) The method of claim 1, wherein the range of addresses ~~invariant property~~ is a range of instruction addresses ~~and further comprising verifying that the runtime value of the data structure is within the range of instruction addresses specified in source code of the computer program~~.

5. (currently amended) The method of claim 1, wherein the range of addresses ~~invariant property~~ is a range of instruction addresses and further comprising:

automatically generating during compilation a valid instruction address range including an upper bound and a lower bound for the range of addresses, wherein the source code of the computer program does not include a specification of the upper bound and lower bound; and

verifying that the runtime value of the data structure is within the valid instruction address range.

6. (canceled)

7. (original) The method of claim 1, further comprising communicating the invariant property from a compiler to a code generator.

8. (original) The method of claim 7, further comprising storing the invariant property in a symbol table.

9. (currently amended) The method of claim 8, wherein the ~~invariant property~~ range of addresses is a range of data addresses and further comprising ~~verifying that the runtime value of the data structure is within a range of data addresses specified in source code of the computer program~~.

10. (currently amended) The method of claim 8, wherein the ~~invariant property~~ range of addresses is a range of data addresses and further comprising:

automatically generating during compilation a valid data address range including an upper bound and a lower bound for the range of data addresses, wherein the source code of the computer program does not include a specification of the upper bound and lower bound; and

verifying that the runtime value of the data structure is within the valid data address range.

11. (currently amended) The method of claim 8, wherein the ~~invariant property~~ range of addresses is a range of instruction addresses ~~and further comprising verifying that the runtime value of the data structure is within the range of instruction addresses specified in source code of the computer program~~.

12. (currently amended) The method of claim 8, wherein the ~~invariant property~~ range of addresses is a range of instruction addresses and further comprising:

automatically generating during compilation a valid instruction address range including an upper bound and a lower bound for the range of addresses, wherein the source code of the computer program does not include a specification of the upper bound and lower bound; and

verifying that the runtime value of the data structure is within the valid instruction address range.

13. (canceled)

14. (original) The method of claim 8, further comprising storing in the symbol table one or more code addresses associated with one or more updates to the data structure.

15. (currently amended) An apparatus for verifying at runtime an invariant property of a data structure of a computer program, comprising:

means for automatically generating a first code segment that verifies a runtime value of the data structure is consistent with the invariant property in response to an annotation of the data structure that defines the invariant property of the data structure,

wherein the invariant property is a range of addresses specified in the annotation in source code of the computer program;

means for comparing the runtime value of the data structure with the invariant property during execution of the program via execution of the first code segment and for verifying that the runtime value of the data structure is within the range of addresses; and

means for performing a programmed action if the runtime value is inconsistent with the invariant property.

16. (currently amended) A computer-implemented method for verifying at runtime an invariant property of a data structure of a computer program, comprising:

determining an invariant property of a data structure from a source code specification of the data structure and an associated specification of the invariant property in the source code, wherein the specification of the invariant property defines the invariant property without checking whether a variable used with the data structure is consistent with the invariant property, and the invariant property is a range of addresses specified in the annotation in source code of the computer program;

generating from the specification of the invariant property a first executable code segment that determines whether a value of a variable used with the data structure is consistent with the invariant property;

determining during execution of the first executable code segment whether the value of the variable used with the data structure is within the range of addresses ~~consistent with the invariant property~~; and

performing a programmed action in response to the value of the variable being outside the range of addresses ~~inconsistent with the invariant property~~.

17. (currently amended) The method of claim 16, wherein the ~~invariant property~~ range of addresses is a range of data addresses established during compilation of program code that instantiates the data structure ~~and the step of determining whether the value of the variable is consistent with the invariant property includes determining whether the value of the variable is within the range of data addresses~~.

18. (currently amended) The method of claim 16, wherein the ~~invariant property~~ range of addresses is a range of data addresses and further comprising:

automatically generating during compilation a valid data address range including an upper bound and a lower bound for the range of data addresses, wherein the source code of the computer program does not include a specification of the upper bound and lower bound; and

determining whether the value of the variable used with the data structure is within the valid data address range.

19. (currently amended) The method of claim 16, wherein the ~~invariant property~~ range of addresses is a range of instruction addresses established during compilation of program code that instantiates the data structure and further comprising verifying that the value of the variable used with the data structure is within the range of instruction addresses specified in source code of the computer program.

20. (currently amended) The method of claim 16, wherein the ~~invariant property~~ range of addresses is a range of instruction addresses and further comprising:

automatically generating during compilation a valid instruction address range including an upper bound and a lower bound for the range of addresses, wherein the source code of the computer program does not include a specification of the upper bound and lower bound; and

determining whether the value of the variable used with the data structure is within the valid instruction address range.

21. (canceled)

22. (previously presented) The method of claim 16, further comprising communicating the invariant property from a compiler to a code generator.

23. (previously presented) The method of claim 22, further comprising storing the invariant property in a symbol table.

24. (currently amended) The method of claim 23, wherein the ~~invariant property~~ range of addresses is a range of data addresses ~~and the step of determining whether the value of the variable is consistent with the invariant property includes determining whether the value of the variable used with the data structure is within a range of data addresses specified in source code of the computer program~~.

25. (currently amended) The method of claim 23, wherein the ~~invariant property~~ range of addresses is a range of data addresses and further comprising:

automatically generating during compilation a valid data address range including an upper bound and a lower bound for the range of data addresses, wherein the source code of the computer program does not include a specification of the upper bound and lower bound; and

determining whether the value of the variable used with the data structure is within the valid data address range.

26. (currently amended) The method of claim 23, wherein the ~~invariant property~~ range of addresses is a range of instruction addresses established during compilation of program code that instantiates the data structure ~~and the step of determining whether the value of the variable is consistent with the invariant property includes determining whether the value of the variable used with the data structure is within the range of instruction addresses specified in source code of the computer program~~.

27. (currently amended) The method of claim 23, wherein the ~~invariant property~~ range of addresses is a range of instruction addresses and further comprising:

automatically generating during compilation a valid instruction address range including an upper bound and a lower bound for the range of addresses, wherein the source code of the computer program does not include a specification of the upper bound and lower bound; and

determining whether the value of the variable used with the data structure is within the valid instruction address range.

28. (canceled)

29. (previously presented) The method of claim 23, further comprising storing in the symbol table one or more instruction addresses at which respective updates are made to the data structure.

30. (currently amended) An apparatus stored on a computer readable medium and executing on a computer for verifying at runtime an invariant property of a data structure of a computer program, comprising:

means for determining an invariant property of a data structure from a source code specification of the data structure and an associated specification of the invariant property in the source code, wherein the specification of the invariant property defines the invariant property without checking whether a variable used with the data structure is consistent with the invariant property, and the invariant property is a range of addresses specified in the annotation in source code of the computer program;

means for generating from the specification of the invariant property a first executable code segment that determines whether a value of a variable used with the data structure is consistent with the invariant property;

means for determining during execution of the first executable code segment whether the value of the variable used with the data structure is within the range of addresses ~~consistent with the invariant property~~; and

means for performing a programmed action in response to the value of the variable being outside the range of addresses ~~inconsistent with the invariant property~~.